约数

2019年2月25日 15:49

-
- 约数

一. 定义和定理
  1. 定义：整数n取余d的余数为0，称d能整除n，称d是n的约数，n是d的倍数，可记为d|n
  2. 若正整数N被唯一分解为i从1到mΣpi^ci（m为质约数个数），则Nd正约数集合为{Σpi^bi}，其中0<=bi<=ci
  3. 因而N的正约数个数为i从1到m п (ci+1)
  4. N的所有正约数的和为i从1到m п j从0到ci Σ pi^j = (1+p1+p2^2...)*...

二. 求正约数的集合
  1. 试除法求n的正约数：循环1~√n
     i.
```
for__(i,1,sqrt(n))
     if(n%1==0){
          factor[ed++]=i;
          if(i!=n/i)
               factor[ed++]=n/i;}
```
  2. 试除法推论：整数N的约数的个数的上界为2√N
  3. 倍数法求1~N的正约数集合：i循环1~n，再j循环1~n，给i的j倍标记上I
     i.
```
vector<int> factor[500010];
for__(i,1,n)
     for__(j,1,n/i)
          factor[i*j].push_back(i);
```
  4. 倍数法的推论：1~N的正约数个数总和约NlogN（调和级数趋向lnN）

三. Pollard-ρ算法分解大数
  1. 大致原理是随机找几个数求差，让差与大数N求gcd，据说复杂度约O(N^{1/4})
  2. Pollard设计的找随机数的方法是令x=x*x+c，会生成环，因此又称ρ形找环法
  3. 例：洛谷P4718求最大质因子，顺便用了下Miller Rabin判质数

```
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <random>
#include <algorithm>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef long double lb;

ll m, ans;   //对每个m，将其最大质因子存进ans

mt19937_64 rnd(time(0));  //c++11的std的魔法，下文可用rnd()生成[0,2^64)

ll read(){
     ll k=0; int f=1;
     char c=getchar();
     while(!isdigit(c)){ if(c=='-') f=-1; c=getchar(); }
     while(isdigit(c)) k= k*10 + c-48, c=getchar();
     return k*f;
}
```

```
inline ll Abs(ll x){ return x<0 ? -x : x;}    //取绝对值

inline ll qmul(ull x, ull y, ll p){    //O(1)x*y%p
    return (x*y - (ull)((lb)x/p*y)*p + p)%p;
}

inline ll qpow(ll x, ll y, ll p){      //x^y%p
    ll res=1;
    for(; y; y>>=1, x=qmul(x,x,p))
        if(y&1) res=qmul(res,x,p);
    return res;
}

inline bool MR(int tc, ll p){    //miller rabin判质数，用tc检测p是不是质数

    if(qpow(tc,p-1,p)!=1) return 0;  //费马小定理
    ll y=p-1, z;
    while(!(y&1)){      //二次探测
        y>>=1;
        z=qpow(tc,y,p);
        if(z!=1 && z!=p-1) return 0;
        if(z==p-1) return 1;
    }
    return 1;
}

inline bool isPrime(ll x){//用5个小质数做MR测试，返回n是质数?1:0
    if(x<2) return 0;
   if(x==2 || x==3 || x==5 || x==7 || x==43) return 1;
   return MR(2,x) && MR(3,x) && MR(5,x) && MR(7,x) && MR(43,x);
}

inline ll rho(ll p){       //玄学求出p的非平凡因子
    ll x,y,s,c;  //s用来存（y-x）的乘积
    for(;;){      //求出一个因子来
        y=x=rnd()%p;
        c=rnd()%p;
        s=1;
        int i=0, I=1;
        while(++i){  //开始玄学倍增生成
            x=(qmul(x,x,p)+c)%p;       //以平方再+c的方式生成下一个x
            s=qmul(s,Abs(y-x),p);      //将每一次的（y-x）都累乘起来
            if(x==y || !s) break;      //换下一组，当s=0时，继续下去是没意
            义的
            if(!(i%127) || i==I){      //每127次求一次gcd，以及按j倍增的求
            gcd
                ll g=__gcd(s,p);
                if(g>1)      return g;
                if(i==I) y=x, I<<=1;       //维护倍增正确性，并判环
            }
        }
    }
}

inline void PR(ll p){      //找p的最大质因子，存进全局变量ans
    if(p<=ans) return ; //最优性剪枝
    if(isPrime(p)) return ans=p, void();
```

```
    ll pi=rho(p);//玄学求出任一因子p_i
    while(p%pi==0) p/=pi;
    return PR(pi), PR(p);      //向下分治
}

int main(){
    int n=read();
    for(int i=0; i<n; ++i){
        ans=1;
          PR( m=read() );
        if(ans==m) puts("Prime");
        else printf("%lld\n",ans);
    }
    return 0;
}
```

◆ 公约数

四. 公约数common divisor

1. 公约数：同时是自然数a和b的约数的自然数d即为a和b的公约数，最大的公约数记为gcd(a,b)
2. 公倍数common multiple：同时是自然数a和b的倍数的自然数m，最大的公倍数记为lcm(a,b)
3. 定理：gcd(a,b)*lcm(a,b)=a*b
   i. 推论：lcm(a,b)=a*b/gcd(a,b)
   ii. 引理1：令g=gcd(a,b)，a0=a/g，b0=b/g，易知a0b0互质
   iii. 引理2：互质数的gcd是1，lcm是其乘积
   iv. 引理3更相减损：gcd(a*g,b*g)=gcd(a,b)*g
   v. 引理4：lcm(a*g,b*g)=lcm(a,b)*g
   vi. 推论证明：lcm(a,b)=lcm(a0,b0)*g=a0*b0*g=a*b/g

五. 最大公约数

1. 欧几里得辗转相除算法
```
int gcd(int a,int b){ return b? gcd(b, a%b): a; }
```
   i. 证明：a<=b时显然成立；a>b时令a=q*b+r，d|a且d|b则d|a-q*b即d|r，则d既是ab公约数又是br公约数
   ii. 例：gcd(a,a)=**gcd(0,a)=a**
   iii. 因此for[i=0:n-1] gcd(i,n) = for[i=1:n] gcd(i,n)
2. 更相减损法
```
int gcd(int a,int b){ return b?gcd(max(a-b,b),min(a-b,b)): a; }
```
   i. 需要判断a和b哪个大，只适用于精度不方便取余的情况
   ii. 证明：d|a且d|b则d|a-b
3. 求最小公倍数
```
int lcm(int a,int b){ return (ll)a*b/gcd(a,b); }
```
4. <algorithm>还是GNU编译器，总之有个__gcd模板，自动算相同类型的两数的gcd，返回值也是该类型
5. Bézout裴蜀（贝祖）定理：ax+by能=c iff c是gcd(a,b)的倍数（abc正整数）（xy是整数）
   i. 证明：a和b均为gcd(a,b)的整数倍，因此ax+by也是gcd(a,b)的整数倍

- ◆
- ◆ 例题

六. 公约数公倍数

1. 求n个数可组合出的数在k进制下能取到的所有余数（CF1010C）
   i. 由贝祖定理，能组合出的数一定是n个数和k本身的gcd的倍数
2. 求不大于n的正整数中是a或b的倍数的数据量
   i. 思路：a的倍数+b的倍数-ab共同的倍数
   ii. 共同的倍数不一定是指a*b的倍数，而是lcm(a,b)的倍数
   iii. 因为是<=，所以可以直接用整除
   iv. 输出n/a+ n/b- n/lcm(a,b)即可
3. 求gcd是x，lcm是y的无序数组(p,q)的个数
   i. 思路：枚举x的每个<=y的倍数p，利用p*q=x*y反求q，若gcd为x，反带回去发现lcm也是y
   ii.
```
ULL M=x*y,m=y/x;
for__(i,1,m){
	p=i*x;  //确保p的约数有x，提高效率
	if(y%p)  //确保p的倍数有y，避免下一步的整除带来错误
		continue;
	//q=M/p;          //p*q==gcd(p,q)*lcm(p,q)
	if(gcd(p,M/p)==x)
		++cnt;}
```
4. 求1~n中任三个数的lcm的最大值
   i.
```
if(n<3){
	cout<<n;
	return 0;}
if(n&1)              //奇数偶数奇数一定互质
	cout<<n*(n-1)*(n-2);
else{
	if(n%3)  //此时跳过n-2就能保证没有3这个公约数
		cout<<n*(n-1)*(n-3);
	else      //很遗憾，此时不能选n，可以看具体样例
		cout<<(n-1)*(n-2)*(n-3);}
```

七. 公约数公倍数应用

1. 求含有n*m个方格的表格的顶点能组成的三角形个数
   i. 通过枚举能覆盖该三角形的最小矩形，来达到完整计数
   ii. 易知这种计算法和矩形位置无关，每个i*j的矩形有(n-i+1)*(m-j+1)种
   iii. 最小覆盖iff至少有一个三角形顶点在矩形顶点上
   iv. 只有一个顶点已知在矩形格点，则另两个动点的位置有(i-1)*(j-1)种
   v. 只有两个顶点已知在矩形非对角格点上，则另一个动点有i-1或j-1种
   vi. 只有两个顶点已知在对角格点上时，第三个动点几乎可以出现在所有(i+1)*(j+1)种点上，除了所在直线所通过的格点gcd(i,j)+1个（其中有两个是已知顶点）再除了两个其他格点，即(i+1)*(j+1)-gcd(i,j)-3种
   vii. 三个顶点都固定在已知对角格点上，1种
   viii. 以上情况的固定方法分别有4，2，2，4种，求和，得6* i* j- 2gcd(i,j)

ix. for__(i,1,n) for__(j,1,m)

```
ans+= (n-i+1)* (m-j+1)* (6*i*j- 2*__gcd(i,j));
```

x. 或先赋初值c(n*m, 3)再删去共线的数量，删法可以用枚举向量，再计算平移的方法数

xi.
```
t= (n+1)* (m+1);
ans= t* (t-1)* (t-2)/ 6;
for__(i,0,n) for__(j,0,m)
    if (i || j)
        if (!i || !j) ans-= (ll) (gcd(i,j)-1)* (n-i+1)* (m-j+1);
        else ans-= (ll) 2* (gcd(i,j)-1)* (n-i+1)* (m-j+1);
```

2. 对整数列各乘以可以不相同的任意整数，求新数列绝对值最小的和（P4549）
   i. 由贝祖定理，对数列取绝对值后求gcd即可

八. 因数

1. 递归求前n个整数的最大奇因数的和
   i.
```
ll ans(ll n){
    if(n==1)//终点
        return 1;
    if(n&1)         //f(奇数n)= 奇数n
        return n + ans(n-1);
    //else          //f(偶数n)= f(偶数n/2)
    //ans(偶数n)= sum{f(奇数)}+ sum{f(偶数)}
    //        = 1+3+……+n-1 + sum{f(2/2)+……f(n/2)}
        return n* n/2 /2 + ans(n/2);}
```

2. 在[a,b]中找x，[c,d]中找y，求x*y是2018的倍数的可能数

```
int m[4]={1,2,1009,2018};       //2018的所有因子

void init(int *x,int a,int b){
    for_(i,0,4)       //找[a,b]中m[i]的整数倍的个数
        x[i]=b/m[i]-(a-1)/m[i];
    x[2]-=x[3];       //去重
    x[1]-=x[3];
    x[0]-=x[1]+x[2]+x[3]; }

int a,b,c,d;
scanf("%d%d%d%d",&a,&b,&c,&d);
int x[4],y[4];     //m[i]的非m[1~4]倍数，在区间内的个数
init(x,a,b);
init(y,c,d);
ULL ans=0;
for_(i,0,4)
    for_(j,0,4)
        if(m[i]*m[j]%2018==0)
            ans+=(ULL)x[i]*y[j];
cout<<ans;
```

3. 输出满足a^n+b^n=c^n的b和c，没有的话输出-1 -1
   i.
```
if(n==0 || n>2){//1+1=1，费马大定理
    printf("-1 -1");
    return 0;
}
if(n==1)
```

```
        b=1,
        c=a+1;
else    //n==2
        if(a&1)
                b=a*a/2,
                c=b+1;
        else
                b=a*a/4-1,
                c=b+2;
printf("%llu %llu",b,c);
/*      n为2时，利用平方差公式，a*a=(b+c)*(b-c)
        a为奇数时，易知b-c=1时有整数解，此时b=a*a/2
        a为偶数时，易知b-c=2时有整数解，此时b=a*a/4-1
        这个好像叫奇偶数列法则
*/
```

4. 组合数N C m的因子个数
    i. 将阶乘拆开来做，求每个数的质因子，记录每个质因子出现的幂次数
    ii.
```
vector<int> k[MN];      //各个数的质因数分解
ll ans[MN];
cin>>n>>m;
if(m>n/2)
        m= n-m;
for__(i,1,n){
        int t= i;
        for__(j,2,i)
                while(t>1 && t%j==0){
                        k[i].emplace_back(j);
                        t/=j;}}
for_(i,0,m){
        for(auto t: k[n-i])
                ++ans[t];
        for(auto t: k[i+1])
                --ans[t];}
ans[0]= 1;
for__(i,1,n)
        ans[0]*= (ans[i]+1);
cout<<ans[0];
```

5. 递归求分解因数方法数（注意去重，可以理解为不减序列数）
    i. 法一：从大到小循环查x的因数数量，找到一个因数后再查因数的因数
    ii.
```
int f(int x, int a){
        if(x==1) //不可分解了
                return 1;
        if(a==1) //边界
                return 0;
        if(x%a==0)      //分解新找到的因数
                return f(x,a-1) + f(x/a,a);
        //else  //找下一个因数
                return f(x,a-1);}
```
    iii. 法二：每找到一个因数（或因数的因数）给ans+1
    iv.
```
void f(int x, int a){
        if(x==1){
```

```
        ++ans;
        return;
}
for__(i,a,x)
        if(x%i==0)
                f(x/i,i);}

cin>>n;
while(n--){
        cin>>x;
        f(x,2);
        cout<<ans<<endl;}
return 0;}
```